International Journal of Computer Science Engineering
and Information Technology Research (IJCSEITR)
ISSN(P): 2249-6831; ISSN(E): 2249-7943
Vol. 5, Issue 1, Feb 2015, 99-110

A PROPOSED SECURITY FRAMEWORK FOR NETWORK MANAGEMENT BASED ON COMMON OBJECT REQUEST BROKER ARCHITECTURE (CORBA) USING MOBILE AGENT

RAJESH KUMAR[1] & S NIRANJAN[2]

[1]Research Scholar, Mewar University Rajasthan, India

[2]Professor, Department of Computer Science & Engineering, GITAM, Kablana, Jhajjar, Haryana, India

ABSTRACT

In this paper we are going to introduce security framework for network management using mobile agent based on CORBA. Security is a major concern for any type of communication system. Here we are introducing Operational and Secure Environment that plays a vital role in network management domain for Mobile Agents. The architecture proposed by us consists of Security Interfacing (SI), Agent Communication Manager (ACM), Agent Manager Skelton, Portable Object Adaptor (POA), Appellative Service Manager, Object Implementation and ORB Interface. Functionality for all of them works together to achieve the task of secure network management using mobile agents. The architecture also supports the heterogeneous network management philosophy. The proposed security framework emphasises on secure network management adaptability. This framework besides communicating among devices, ensure the communication flow and provides interaction amongst them. The security feature is additionally augmented over the already existing architecture for network management based on CORBA.

KEYWORDS: Common Object Request Broker Architecture (CORBA), Security Interfacing (SI), Security Application Interfacing (SAI), Agent Communication Manager (ACM), Portable Object Adaptor (POA), Object Request Broker (ORB), Service Control Node (SCN), Service Switching Point (SSP)

INTRODUCTION

Mobile Agent can be simply thought of as an entity which can run in a dynamic environment with autonomous ability and mobility.

Overview of Mobile Agent Architecture

Figures 1 to 5 show the organisation of mobile agent systems. Mobile agent travels in and around network environment visiting processing elements, hopping from one host to another [3]. The execution of the program code and the dispatching agents to different computing nodes is handled by mobile agent servers. Each agent has its own thread, which is executed by host server. The communication between various servers or agents is performed by messages transfer mechanisms.

Mobile Agent Interaction on a Server

Figure 1 shows the relationships between various agents on a given server as they communicate using messages to accomplish the execution tasks. The resources needed by the agents while visiting the host are allocated by the host itself

as described in the subsequent sections.

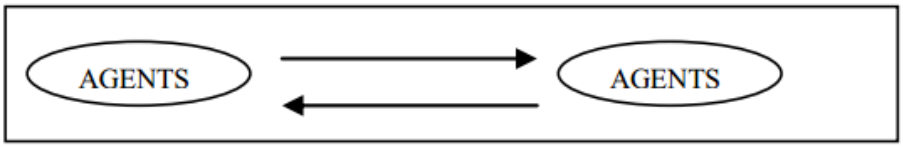

Figure 1: Agent on Servers

Mobile Agent Server Architecture

Figure 2 shows the block diagram of a mobile agent server. The server coordinates the activities of the visiting agents. Message transfer from an important component in the mobile agent communication.

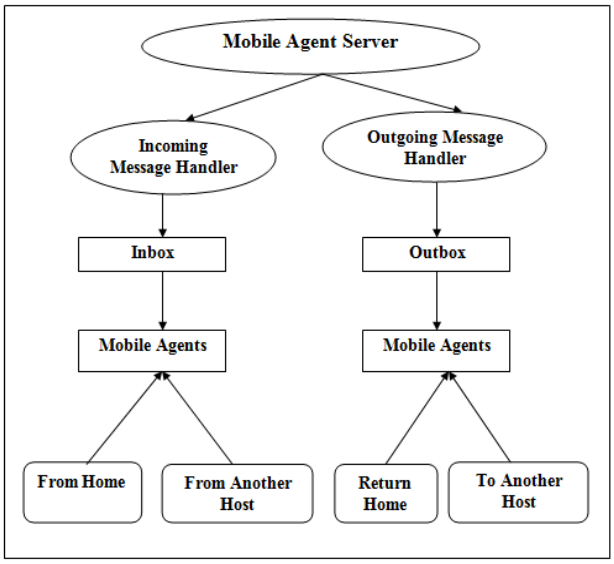

Figure 2: Mobile Agent Server Architecture

Transfer of Mobile Agents between Servers

When an agent completes its task on a host, it either migrates to another host platform or returns to its home host. Figure 3 shows how agents are dispatched from one host to another using object serialization. Object serialization is a process of converting a data structure or object into format that can be stored and retrieved later in the same or another computing environment [3]. When the resulting series of bits is read according to the serialization format, it can be used to create a semantically identical clone of the original object.

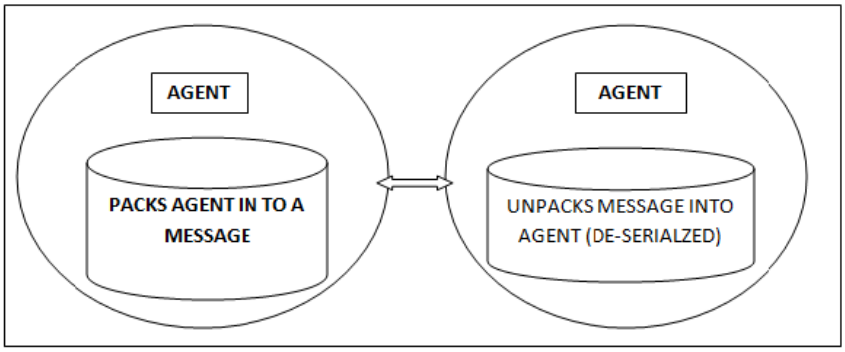

Figure 3: Transfer of Mobile Agents between Server

Message Exchange between Agents on the Same Server

The host platform facilitates exchange of messages between two or more agents on the host and between different hosts. Figure 4 show the role of the server in intra/inter server agents' communication and the Figure 5 illustrates message exchange between on different servers.

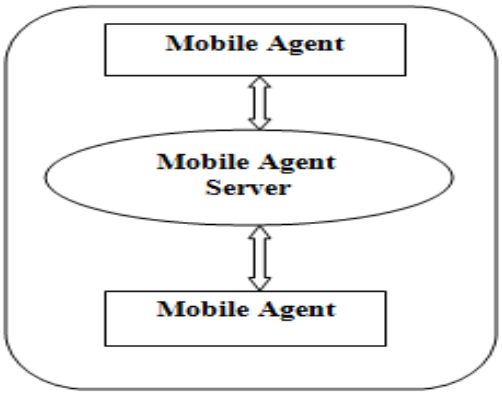

Figure 4: Message Exchange between Agents on the Same Server

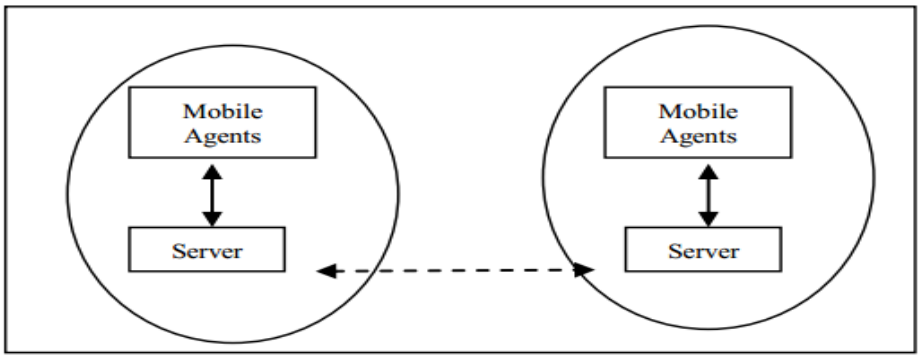

Figure 5: Message Exchange between Agents on different Servers

Agent Migration between Intra and Inter Network Paradigms

The following entities are required to allow agents to migrate across a network or to another network. [3]

- Common execution language
- Process persistence
- Communication mechanism between agent hosts
- Security to protect agents and agent hosts

COMMON EXECUTION LANGUAGE

When a process is to migrate from one host to another, both hosts share a common execution language for a homogenous networking environment. It's conceivable that assembly language or machine code could be sent across the network for execution. But in the case of heterogeneous environment, where many different system architectures are connected, interpreted scripting language or emulation of a system that is capable of executing machine code [4] can be used to solve the problem of a common execution language**.**

PROCESS PERSISTENCE

For processes which wants to migrate from current machine to remote machines, they must be capable of saving their execution state, or spawning (to create) a new process whose execution state will be saved. This property is called persistence. Persistence involves converting the object's state (variables, stack, and possibly even the point of execution) and converting it into a data format suitable for transmission over a network.

COMMUNICATION MECHANISM BETWEEN AGENT HOSTS

An agent can be transferred using TCP/IP, or by using a higher level of communication such as RMI, IIOP, SMTP or even HTTP. Mobile agent architectures may even use a variety of transport mechanisms, giving greater flexibility.

SECURITY TO PROTECT AGENTS AND AGENT HOSTS

Security is very important issue when the executable code is transferred across a network or within a network. Encryption and decrypted technique is used. Other technologies such as digital signature will be used. The use of digital signatures, the identity of an agent and its user can be authenticated [2]**.**

Technologies Used for Mobile Agent Network Management and Agent Migration

- **Message Passing Systems**

Software agents need not always travel across a network to communicate with information sources, or other agents. One of the most important message passing systems for agents is the Knowledge Query Manipulation Language (KQML). KQML is an important mechanism for communication, because it allows much more complex forms of interaction than query/response mechanisms. KQML [6] allows agents to communicate using a rich set of messages called per formatives, and is capable of communicating information about attributes, rather than just data and facts. Message passing systems like KQML don't require mobility; they can simply pass a message and have it delivered through some transport mechanism.

Remote Method Invocation (RMI)

Remote method invocation allows Java developers to create distributed systems that share objects. New objects can be transferred across the network, and RMI is becoming a popular mechanism for agent communication. RMI can be used to facilitate mobility (as the mechanism for transport), or as a replacement that allows agents to invoke methods of other agents [6]**.**

Common Object Request Broker Architecture (CORBA)

CORBA is a platform and language independent mechanism for invoking remote object methods. Unlike RMI which is specific only to Java Virtual Machines, CORBA can be used to create distributed systems that execute on many platforms, in many languages. CORBA holds great potential, because of its portability and flexibility [9]. CORBA is a direct threat to mobile agency, and would allow developers to create agents that are capable of complex communication without ever travelling across a network.

Here Emphasis is laid on the CORBA technique for Network Management and the key points are as follows:

- Used for Agent Migration

- Simple Network Management
- Distributed Network Management
- Supports Single Agent
- Support Multi Agent

PROPOSED SECURITY FRAMEWORK FOR NETWORK MANAGEMENT BASED ON CORBA USING MOBILE AGENT

CORBA is the combination of Architecture Object Model plus Reference Model. The Object Model Shows how objects can be described in the context of a Heterogeneous distributed Environment while the reference model describes the possible interaction between Distributed Objects. The generalised architectures studied like Mole, D'Agent, and Aglet etc, require additional features to ensure security. The suggested architecture accommodates the essential conditions for ensuring security [7], which supports the secure distributed Network Management as well as operates on Heterogeneous network Components. This Architecture is based on CORBA [8]. Here Operational Environment is an Execution Environment for Mobile Agents. In Agent based Operational Environment there is different components like Agent Communication Manager, Agent Manager Skelton, Object Implementation Orb Interface, Object Adaptor and Portable object Adaptor each have its own task to perform.

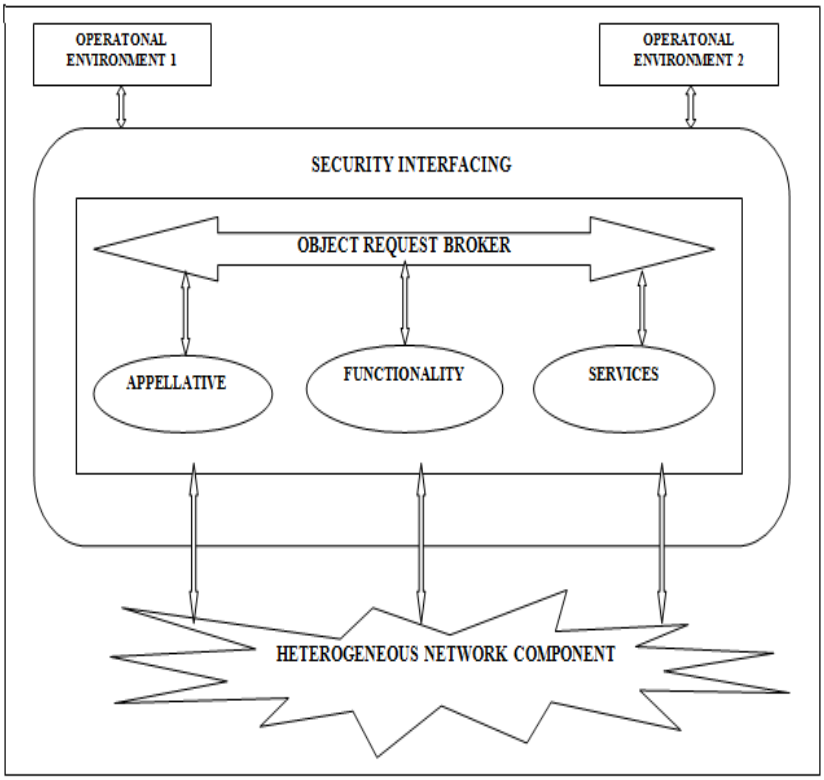

Figure 6: Proposed Security Architecture

In the above design each agent not only has the ability of conducting specific Network Management task but also implements the Behaviour that is common to all agents. The Operational Environment interacts with ORB (Object Request Broker) interface through Agent Manager Skelton, Via POA (Portable object Adaptor).

Workflow of Proposed Security Framework

The proposed security framework will work in the following manner:

The Operational Environment initially interacts with the Security Interface and Security Interface provides the mechanisms of the next work flow. It will further work with Security Application Interface and the security interface ORB. The Operational Environment will be compatible with any device on the network like Desktop, Laptop, PDA or Mobile hand held device etc. This security framework also works with the Heterogeneous network component.

Security Interfacing

Security interfacing in CORBA includes interfaces that define services for the following well-known areas of computer security: Authentication, Message Protection (including encryption for guaranteeing confidentiality as well as integrity), Access Control, Auditing, and Non-repudiation. The latter two provide means to achieve some degree of accountability [4]. In addition, CORBA Security describes interfaces that can be used for coping with the tedious, but very crucial task of security management/ administration (e.g. assigning policies to domains). Furthermore they take care of specific problems with object-oriented (security) systems, such as the delegation of rights.

In order to meet its objectives, CORBA Security interfacing design is based on some general principles. The most important ones are transparency, scalability, flexibility, and inter operability, presented in the following [11]. The functionality of Security interfacing is provided in three distinct levels of increasing functionality. The bottom Level 0, which is not part of the specification in the beginning, but was added later on, just integrates SSL into CORBA. Level 1, provides all the above mentioned security measures except for non-repudiation. This is done in such a way that the applications running on top of the CORBA system are not aware of the security features provided underneath. Level 1 thus covers the security unaware applications.

On the contrary, Level 2 deals with applications those are security aware and are consequently in a position to interact with CORBA Security in order to specify the exact security features used. For these security aware applications non-repudiation is also relevant.

SECURITY INTERFACING DOES THE FOLLOWING FUNCTION

- Transparency: application-level objects should be unaware of security services which are used.
- Control: client/object should be able to specify security requirements
- Security polices: specified by policy objects
- Administrative domain where client/server is executed determines set of security services.

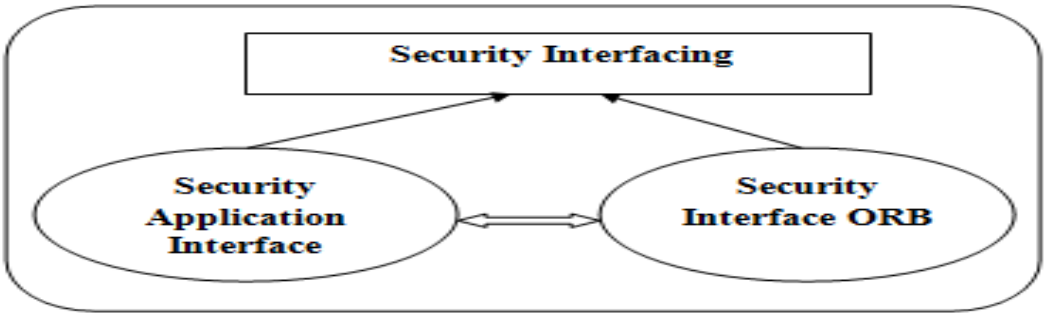

Figure 7: Security Interfacing

Object Request Broker

In distributed computing, an Object Request Broker (ORB) is a piece of middleware software that allows programmers to make program calls from one computer to another via a network.[9] ORBs promote interoperability distributed object systems because they enable users to build systems by piecing together objects from different vendors, so that they communicate with each other via the ORB. ORBs handle the transformation of in-process data structures to and from the byte sequence, which is transmitted over the network.

This is called marshalling or serialization. Some ORBs, such as CORBA-compliant systems, use an Interface Description Language (IDL) to describe the data that is to be transmitted on remote calls. In addition to marshalling data, ORBs often expose many more features, such as distributed transactions, directory services or real-time scheduling. In object-oriented languages, the ORB takes the form of an object with methods enabling connection to the objects being served. After an object connects to the ORB, the methods of that object become accessible for remote invocations. The ORB requires some means of obtaining the network address of the object that has now become remote. The typical ORB also has many other methods [9].

Services to Client Objects by ORB

The following services are offered by ORB to Client-Objects [8]:

OBJECT LOCALIZATION

The client does not need to know if the target object is active either within a process running on a remote machine or in a process within the same machine or even within the same process of the client object implementation. The client must not be aware of the programming language used to implement the target object, of the operating system or the hardware platform on which the server program is executed.

OBJECT EXECUTION STATE

The client does not need to know if the target object is already active within the corresponding server process and ready for accepting the incoming request. The ORB is responsible for transparently activating the target object before forwarding the client request.

OBJECT COMMUNICATION MECHANISM

The client can ignore the communication mechanism used by the ORB to send the incoming request to the target object and to return the response to the client.

Operational Environment

Operational Environment Consist of Agent Communication manager, Agent Manager Skelton, Object implementation, Static and Dynamic Skelton, ORB Interface [8], Object Adaptor, Portable Object Adaptor, Java Virtual Machine can run on Operational Environment.

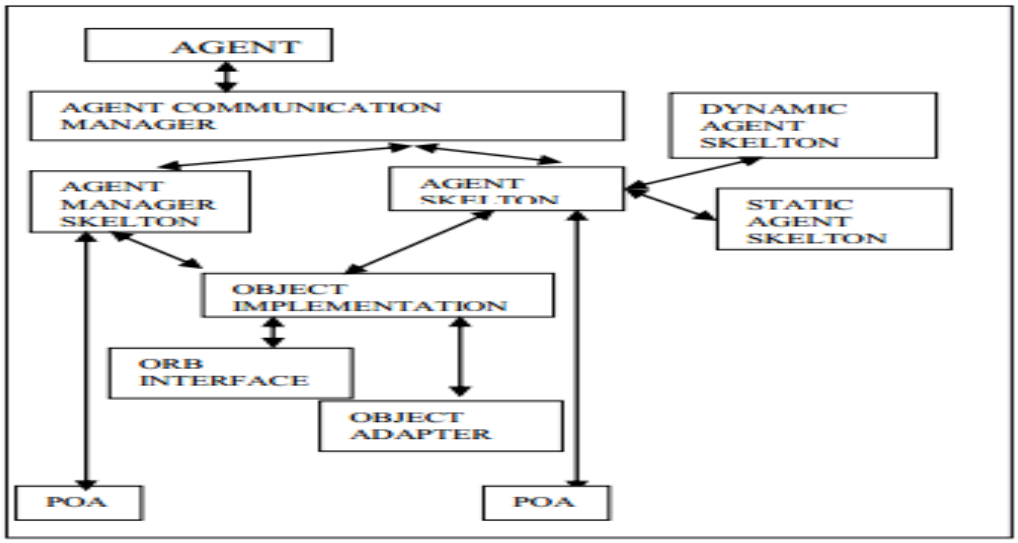

Figure 8: Working Diagram of Operational Environment

Agent Communication Manager

The task of creating, transferring, and receiving, suspending and terminating agent has been handled by Agent Communication Manager. It transfers the information to the Agent Manager Skelton if it is a Single Agent, otherwise it goes to agent Skelton. It checks in the first step whether it is static Agent or Dynamic Agent Skelton. Then the task for object implementation comes on for ORB Interface and object Adaptor for POA [1]. This Operational environment also works on the Single, Multi Agents, as well as the Distributed Environment.

APPELLATIVE

The Appellative Service Supports a Hierarchical Name space so the Agent Name system could be organized like a well known Internet name system. Every Domain is identified by a unique domain name and in addition to this each Agent have an Agent Name, Expressing its functionality.

Functionality

It is defined as a Process of transfer of information from one current System to the Remote System.

Service

There are several services provided by our Proposed Architecture these are as follows Agent Transport, Agent Control Service, Agent Communication service, Service Creation Environment, SMS SSP, and SCN etc [3].

Heterogeneous Network Component [7]

It includes Bluetooth Wireless Sensor Network, Infrared which has its own protocol. For Example, in Bluetooth we use the BDAN (Blue tooth Device for Agent Network).

SECURITY APPLICATION INTERFACE

Secure Server Location

It provides the following functions:

- It works under the standard that is developed by the Object Management Group (OMG) and is essential for Agent communication as well as for the security concern architecture.
- It provides the mechanism of information, independent of hardware platforms, programming languages and operating systems.
- It also covers the distributed objects to communicate with one another, whether locally or on remote devices, written in different languages, or at different locations on a network.
- One can share ones location with other agents or can say the group of agents.
- See location [10].
- Ones location is sent from his device only when a friend requests to see his location or if he chooses to send his current location within a message. It is not transmitted or recorded at any other time.
- One can share his location with friends and see his friends on a map.

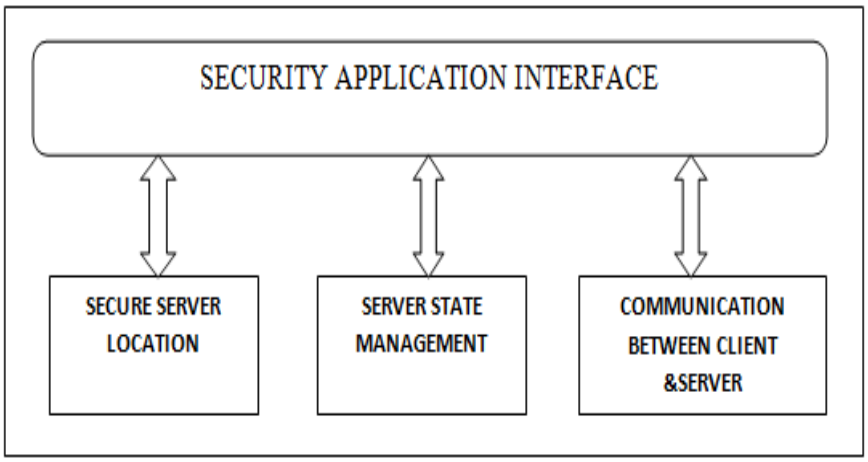

Figure 9: Security Application Interfacing

Server State Management

- It provides the benefit status of communication.
- The communication stage involves various steps with the help of Server State
- Management we can track out the process easily.
- The traffic of network can analyse.
- Active mobile agents on network.
- Provides a mechanism for the status of server whether it is busy or idle.
- From this we can analysis how many processes running in the server.

Communication between Client and Server

- Security Application Interface receives the client request.
- Server confirms the request.
- Server sends reply.
- Client gets reply from server.

Benefits of Security Application Interface

- It offers the system a security mechanism to verify the coming agents.
- Controls the agent functionality.
- Fault tolerance and Agent migration.
- Manages traffic load.
- Improves the system performance [10].

SECURITY INTERFACE ORB

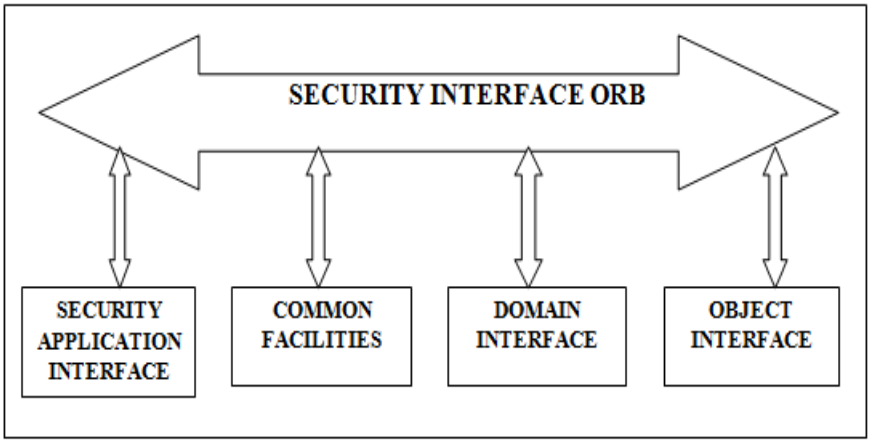

Figure 10: Security Interface ORB

Security Interface ORB works with or we can say act like an interface among the following:

- Security Application Interface
- Common Facilities
- Domain Interface
- Object Interface [5]

Security Application Interface

- It provides the application level interface where authentication wills works.
- Without this we cannot interact with the system

- First Screen that Comes on the Authentication Plays Very Important Role in the Security.

Common Facilities

- It includes the mechanism of entering the authentication
- Password retrieval
- Login facilities
- Different domain interaction etc.

Domain Interface

- Provides the mechanism for server communication
- Facilities to switch various domains
- Listing of Domains.
- Active domains [11]

Object Interface

- It provides the server location
- Server state management
- Communication Process**.**

CONCLUSIONS

The Proposed security framework will support Single Agent, Multi Agent and Distributed Network Management System with improved level of security and Heterogeneous Network component system. The proposed security application interfacing satisfies the task of secure information transfer from one system to another. The Secure Architecture Proposed by us covers the management of large scale networks. The Distributing Management task also reduces the loads of the managed Resources. From the user point of view the security architecture puts only minimum requirement on the computing platforms at the customer site. However before such an approach can be put into application we still have a lot of things to do such as more advancement in the security Mechanisms as the Direction of Future Research.

REFERENCES

1. Yashpal Singh, Rajesh Kumar, S. Niranjan "A Proposed Architecture for Agent Communication Manager in Mobile Ad-hoc Network" IJECS volume 1, Paper ID IJECS, PP 111-115, 2011.
2. Ibharalu Friday Thomas, Sofoluwe Adetokunbo Babatunde, Akinwale Adio Taofiki "A Reliable Protection Architecture for Mobile Agents in Open Network Systems" International Journal of Computer Applications (0975 – 8887) Volume 17– No.7, March 2011.
3. Y Singh, S Niranjan "A Proposed Architecture for Network Management in Mobile Agent Based on Common Object Request Broker Architecture (CORBA)" IFRSA's International Journal of Computing, Vol.2, Issue 1, January 2012.

4. Yashpal Singh, Rajesh Kumar, S. Niranjan "Review of Security Issues for Mobile Agent Technology" IJERD volume 7, ISSUE 4, PP 85-89, May, 2013.
5. Rajesh Kumar, S. Niranjan "A Proposed Security Framework Module of Agent Communication Manager for Mobile Ad Hoc Network" Paper ID: OCT14864, IJSR volume 3, Issue 11, PP 85-89, Nov., 2014.
6. http://www.davidreilly.com/jcb/articles/decompilers_friend_or_foe.html
7. Paolo Bellavista, Thomas Magedanz, Middleware Technologies: CORBA and Mobile Agents, Chapter 5, Dip. Elettronica, Informatica e Sistemistica, Università di Bologna Viale Risorgimento, 2-40136 Bologna –ITALY.
8. www.corba.org
9. http://en.wikipedia.org/wiki/Object_request_broker.
10. www.java.sun.com/docs/books/security/index.html.
11. 2000 Springer-Verlag, Informatik aktuell, http://www.springer.de/comp-de/inf akt/index.html.